Towards Integrating Hybrid DAEs with a High-Index DAE Solver

Nedialko S. Nedialkov, Nacim Ramdani

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Towards Integrating Hybrid DAEs with a High-Index DAE Solver

Nedialko S. Nedialkov — Nacim Ramdani

N° 6834

February 2009

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6834--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Towards Integrating Hybrid DAEs with a High-Index DAE Solver

Nedialko S. Nedialkov*, Nacim Ramdani†

Thème SYM — Systèmes symboliques
Équipe-Projet Coprin

Rapport de recherche n° 6834 — February 2009 — 16 pages

Abstract: J.D. Pryce and N.S. Nedialkov have developed a Taylor series method and a C++ package, DAETS, for solving numerically an initial-value problem differential-algebraic equation (DAE) that can be of high index, high order, nonlinear, and fully implicit. Numerical results have shown this method to be efficient and very accurate, and particularly suitable for problems that are of too high an index for present DAE solvers. However, DAETS cannot be applied to systems of DAEs that change at points it time, also called hybrid or multi-mode DAEs. This paper presents methods for extending DAETS with the capability to integrate hybrid DAEs. Methods for event location and consistent initializations are given. DAETS is applied to simulate a model of a parallel robot: a hybrid system of index-3 DAEs with closed-loop control.

Key-words: high-index differential-algebraic equations, hybrid systems, structural analysis, consistent initialization

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and INRIA France

* Department of Computing Software, McMaster University, Hamilton, Ontario, Canada, L8S 4K1, nedialk@mcmaster.ca
† INRIA Sophia Antipolis – Méditerranée, COPRIN project, nacim.ramdani@inria.fr

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Vers l'intégration numérique d'équations algébro-différentielles hybrides avec un solver de DAE à grande valeur d'indice

Résumé : J.D. Pryce et N.S. Nedialkov ont développé une méthode basée sur le dévelopement en série de Taylor et un package C++, DAETS, pour résoudre numériquement le problème de la valeur initiale pour des systèmes algébro-différentiels (DAE) de grand ordre, non linéaires, totalement implicites et dont la valeur d'indice peut être grande. Les résultats numériques ont montré que cette méthode est efficace, très précise, et particulièrement adaptée aux problèmes dont la valeur d'indice est trop grande pour être résolus par les solveurs actuels. Cependant, DAETS ne peut pas être utilisé avec des systèmes de DAE qui changent à des instants donnés, ou plus généralement des DAEs hybrides ou multi-modes.

Ce rapport de recherche présente des méthodes pour rendre DAETS capable d'intégrer numériquement les DAEs hybrides. Des méthodes pour la détection d'événements et la recherche des nouvelles conditions initiales sont aussi introduites. DAETS est utilisé pour simuler le modèle d'un robot parallèle fonctionnant en boucle fermée, écrit sous la forme d'un système DAE hybride de valeur d'indice 3.

Mots-clés : équations algébro-différentielles à grande valeur d'indice, systèmes hybrides, analyse structurelle, initialisation

Contents

J.D. Pryce and N.S. Nedialkov have developed a Taylor series method and a C++ package, DAETS, for solving numerically an initial-value problem differential-algebraic equation (DAE) that can be of high index, high order, nonlinear, and fully implicit. Numerical results have shown this method to be efficient and very accurate, and particularly suitable for problems that are of too high an index for present DAE solvers. However, DAETS cannot be applied to systems of DAEs that change at points it time, also called hybrid or multi-mode DAEs.

This paper presents methods for extending DAETS with the capability to integrate hybrid DAEs. Methods for event location and consistent initializations are given. DAETS is applied to simulate a model of a parallel robot: a hybrid system of index-3 DAEs with closed-loop control.

Key words: high-index differential-algebraic equations, hybrid systems, structural analysis, consistent initialization

1 Introduction

J.D. Pryce and N.S. Nedialkov have developed the theory and implementation of DAETS, Differential-Algebraic Equations by Taylor Series: a C++ solver for initial-value problems for differential-algebraic algebraic equations (DAEs) [1, 2, 3, 4]. It solves numerically a DAE system of the form

$$f_i(\,t,\text{ the } x_j \text{ and derivatives of them}\,) = 0, \quad i = 1, \ldots, n, \tag{1}$$

in terms of the unknown state variables $x_j(t),\ j = 1, \ldots, n$.

Many DAE solvers are limited by the *differentiation index* of a DAE: the number of times the f_i must be differentiated (w.r.t. t) to obtain equations that can be solved to form an ODE system for the x_j; cf. [5, 6]. The method in DAETS—based on Pryce's structural analysis (SA) theory [7] and Taylor series (TS)—does not find high index inherently hard. DAETS applies this SA, uses automatic differentiation (AD) to expand the solution of (1) in TS, and advances it over a range with a variable-step TS method.

DAETS is very effective when high accuracy is required, and at solving DAEs of high index—we have solved artificial DAEs of index up to 47 [3]. It is versatile: higher-order systems do not have to be cast in first-order form; it can solve explicit and implicit ODEs; it can solve purely algebraic problems, by simple or by arc-length continuation [3, 4].

The f_i are assumed sufficiently smooth. They can be arbitrary expressions built from the x_j and t using $+, -, \times, \div$, other standard functions, and the differentiation operator d^p/dt^p. However, the present theory and implementation do not allow (1) to contain branches, for example `if-then-else` statements, and functions such as abs, min, max, and sign. As a consequence, DAETS cannot be applied to dynamic systems modeled with DAEs that change at discrete points in time, also called hybrid DAEs (HDAEs) [8] or multi-mode DAEs [9]. In particular, we are interested in simulating hybrid dynamic systems, whose continuous dynamics is described by DAEs of the form (1).

We shall refer to (1) with sufficiently differentiable f_i as a DAE, and to (1) with branches in problem formulation as a HDAE; it can be viewed as comprising several DAEs. In this paper, we report ongoing work on extending DAETS to integrate HDAEs, where branches may depend on t, the x_j, and derivatives of

them. In our implementation, switching between DAEs occurs when a scalar-valued *event* function changes sign. We may have m such functions, each of the form

$$g_k(\,t, \text{ the } x_j \text{ and derivatives of them}\,), \quad k = 1, \ldots, m \ .$$

Finding a *consistent initial point* can be one of the hardest parts in DAE solving. Computing such is not difficult for DAETS [3]. However, when branches are present, the structure of a DAE typically changes after an event, a t at which a g_k becomes zero. Then we have to find a consistent point and a DAE "simultaneously", since the new DAE typically depends on such a point and vice versa.

In this work, we give an overview of our methods for event location and consistent initialization after an event is located. We illustrate them by simulating a model of a parallel robot[1]: a hybrid system of index-3 DAEs with closed-loop control. In a future work, we shall analyze these methods in detail and describe their implementation in DAETS.

Section 2 presents an overview of DAETS and summarizes some of the theory behind it. In Sect. 3, we describe our methods for locating an event and consistent initialization after an event point is found. In Sect. 4, we give the model of this robot. Numerical results are in Sect. 5, and conclusions are in Sect. 6.

2 The DAETS Solver

We outline Pryce's SA (§2.1), discuss consistent initialization (§2.2), and give a brief summary of the DAETS implementation (§2.3).

2.1 Pryce's Structural Analysis

Given a DAE (1), the steps of this SA are as follows.

1. Form the $n \times n$ signature matrix $\Sigma = (\sigma_{ij})$, where

$$\sigma_{ij} = \begin{cases} \text{order of derivative of } x_j \text{ in } f_i, \text{ or} \\ -\infty \text{ if } x_j \text{ does not occur in } f_i \ . \end{cases}$$

2. Find a Highest Value Transversal (HVT), which is n positions (i, j) in Σ with one entry in each row and column such that $\sum \sigma_{ij}$ is maximized.

3. Find the smallest *problem offsets* c_i, $d_j \geq 0$ satisfying

$$d_j - c_i \geq \sigma_{ij} \text{ for all } i, j = 1, \ldots, n \quad \text{and} \quad d_j - c_i = \sigma_{ij} \text{ on the HVT} \ .$$

4. Form the system Jacobian $\mathbf{J}$, where

$$\mathbf{J}_{ij} = \begin{cases} \dfrac{\partial f_i}{\partial x_j^{(\sigma_{ij})}} & \text{if } d_j - c_i = \sigma_{ij} \\ 0 & \text{otherwise} \ . \end{cases}$$

[1] Developed at LIRMM UMR CNRS 5506, University Montpellier 2.

Example 1 *Consider the simple pendulum,*

$$\begin{aligned} 0 &= f = x'' + x\lambda \\ 0 &= g = y'' + y\lambda - G \\ 0 &= h = x^2 + y^2 - L^2, \end{aligned}$$

which is an index-3 DAE. The dependent variables are x, y, and λ; G (gravity) and L (length) are constants. The signature matrix, offsets, and system Jacobian are

$$\Sigma = \begin{array}{c} \\ f \\ g \\ h \\ d_j \end{array} \begin{array}{c} \begin{array}{ccc} x & y & \lambda \end{array} \\ \begin{bmatrix} 2^\circ & -\infty & 0* \\ -\infty & 2* & 0^\circ \\ 0* & 0^\circ & -\infty \end{bmatrix} \\ \begin{array}{ccc} 2 & 2 & 0 \end{array} \end{array} \begin{array}{c} c_i \\ 0 \\ 0 \\ 2 \\ \end{array} \quad \textit{and} \quad \mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial x''} & 0 & \frac{\partial f}{\partial \lambda} \\ 0 & \frac{\partial g}{\partial y''} & \frac{\partial g}{\partial \lambda} \\ \frac{\partial h}{\partial x} & \frac{\partial h}{\partial y} & 0 \end{bmatrix} .$$

There are two HVTs, which are marked by $*$ and $\circ$.

If $\mathbf{J}$ is nonsingular at a *consistent point*, then the SA succeeds, and the DAE is solvable in a neighborhood of this point [7, 10]. (Here, $\mathbf{J}$ is nonsingular for any x and y, except $x = y = 0$.) This SA also determines a *structural index*, which is an upper bound for the differentiation index, and usually both are of the same value [3, 7].

2.2 Quasilinearity and Consistency

A d_j is the highest-order derivative of x_j. If the derivatives $x_j^{(d_j)}$ occur in a jointly linear way in the DAE, then we call it *quasilinear*. The required initial values (IVs) that must be specified comprise the vector

$$\mathbf{x} = \left(x_1, x_1', \ldots, x_1^{(d_1+\alpha)}; \; x_2, x_2', \ldots, x_2^{(d_2+\alpha)} \; ; \; \ldots \; ; \; x_n, x_n', \ldots, x_n^{(d_n+\alpha)} \right),$$

where $\alpha = -1$ if the DAE is quasilinear, and $\alpha = 0$ otherwise. To be consistent, $\mathbf{x}$ must satisfy the set of equations

$$\mathbf{0} = \mathbf{f} = \left(f_1, f_1', \ldots, f_1^{(c_1+\alpha)}; \; f_2, f_2', \ldots, f_2^{(c_2+\alpha)} \; ; \; \ldots \; ; \; f_n, f_n', \ldots, f_n^{(c_n+\alpha)} \right),$$

where f_i' means $\mathrm{d}f_i/\mathrm{d}t$. In the quasilinear case, an x_j with $d_j = 0$ does not appear in $\mathbf{x}$, and an f_i with $c_i = 0$ does not appear in $\mathbf{f}$. The extra components in the not quasilinear case are needed to ensure local uniqueness of the solution. For more details see [3, 4].

2.3 Brief Summary of DAETS

The user provides a C++ function for evaluating the f_i. DAETS interprets this function in a symbolic manner through operator overloading, finds the offsets of the problem, and determines if it is quasilinear. Then DAETS uses AD [11] to expand the solution to (1) in TS to as many terms as required. We use the FADBAD++ package [12] to generate functions for evaluating Taylor coefficients (TCs) of the equations f_i. Equating these coefficients to zero gives equations

that are solved for the TCs of the solution components $x_j(t)$. The problem offsets prescribe how to organize the computation of TCs [1, 7, 10] for the solution components of (1); that is, what equation to solve and for which TCs of the solution.

DAETS treats computation of consistent data as an optimization problem and passes this to the optimization package IPOPT [13]. The consistent point that is found is the one that is closest to the point given by the user in a least-squares sense.

The main work of DAETS is done by its `integrate` method [4], which advances the solution, step by step, from a computed consistent point at initial t_0, up to a final t_{end}. Each step generates and sums a TS and then does a *projection* stage to preserve consistency within a user-specified accuracy tolerance. In this paper, we assume $t_{\text{end}} > t_0$; DAETS allows $t_{\text{end}} < t_0$.

3 DAETS and Hybrid DAEs

In §3.1, we present our event detection method, and in §3.2 we describe our approach for consistent initialization, after an event is located.

3.1 Locating Events

Assume that we have a DAE at time t and have found a consistent point $\mathbf{x}$ for it at t. On an integration step from t, DAETS selects a trial stepsize h_{trial} (subject to a user-specified tolerance) and computes scaled, by h_{trial}, TCs for each solution component x_j up to order $p + d_j$,[2] where $p > 0$ is the order of the method. Using these TCs and applying AD, we compute TCs for each event function g_k up to order p. We determine a stepsize $h \leq h_{\text{trial}}$ such that the TS approximations for all g_k on $[t, t+h]$ satisfy the same tolerance as the solution. That is, we select a stepsize such that the event functions are monitored with the same accuracy as the solution. Then we rescale the TCs of the g_k, so they are computed with stepsize h.

Denoting by $(g_k)_i$ the ith such TC, we write

$$\widetilde{g}_k(\tau) = \sum_{i=0}^{p} (g_k)_i \, \tau^i \quad . \tag{2}$$

For each k, we search for real roots of $\widetilde{g}_k$ in $(0, 1]$. If no roots are found, the integration continues with the same DAE from $t + h$. If there are real roots in $(0, 1]$, denote by τ^* the smallest such root. Assume that it is a root of $\widetilde{g}_r(\tau)$, for some $r = 1, ..., m$. Let $h^* = \tau^* h$ and $t^* = t + h^*$.

We compute an approximate TS solution at t^* and project it to satisfy the constraints of the DAE. Denote the projected solution by $\mathbf{x}^*$. If $g_r(t, \mathbf{x})\, g_r(t^*, \mathbf{x}^*) < 0$ or $g_r(t^*, \mathbf{x}^*) = 0$, then we accept t^* as an event point.

However, it is possible that our root finding does not find a real root in $(0, 1]$, for any of the $\widetilde{g}_k(\tau)$, but at least one event function changes sign at $(t^*, \mathbf{x}^*)$. For example, if a $\widetilde{g}_k(\tau^*)$ is very close to zero, because of the projection, g_k may change sign at $(t^*, \mathbf{x}^*)$. In this case, we accept t^* as an event point.

[2] A scaled by h TC is of the form $x_j^{(i)}/i!\, h^i$.

Since an event function changes sign at $(t^*, \mathbf{x}^*)$, this point implies a different DAE than the one used to reach $(t^*, \mathbf{x}^*)$. Now, we have to re-initialize the integration, as discussed in the next subsection.

Remark 1 *When searching for roots of (2), we compute an enclosure of this polynomial in an interval-arithmetic sense as*

$$[\widetilde{g}_k([0,1])] = \sum_{i=0}^{p} (g_k)_i \, [0,1]$$

(similarly to the approach in [14]). If $0 \notin [\widetilde{g}_k([0,1])]$*, then* g_k *does not have a real root in* $(0,1]$*; otherwise, we call the* PROOTS *polynomial root-finder [15] to search for real roots of (2).*

3.2 Structural Analysis and Consistent Initialization

DAETS does not require that IVs are consistent: it typically finds such from guesses given by the user, However, consistent initialization needs to be revisited for HDAEs.

Example 2 *As an illustration of the issues involved, consider the first component in (9),* $\Gamma_{0,1}$*, and the conditions in (14). If the IVs for* $q_{d,1}$*,* q_1*,* $\dot{q}_{d,1}$*, and* $\dot{q}_1$ *at* t_0 *are such that* $\Gamma_{0,1}(t_0) > \alpha$*, then the DAE at* t_0 *contains the equation*

$$\Gamma_1 - \alpha = 0 \ . \tag{3}$$

The SA and the computation of a consistent point are done with this DAE; denote it by DAE-A. During these computations, DAE-A is fixed*—the values on which the branches depend on do not change.*

However, the computed "consistent initial point" for DAE-A may result in $\Gamma_{0,1}(t_0) \in [-\alpha, \alpha]$*. Then, the computed values for* $q_{d,1}$*,* q_1*,* $\dot{q}_{d,1}$*, and* $\dot{q}_{d,1}$ *imply another DAE, denote it by DAE-B, containing the equation*

$$\Gamma_1 - \Gamma_{0,1} = 0$$

instead of (3). As a consequence, we have a point that is likely inconsistent with DAE-B (see also Sect. 5).

Hence, for a HDAE, we need to find not only a consistent point at each t^*, but the "correct" DAE and a consistent point for it. This also applies at the initial time, t_0.

With each combination of values (true or false) in the conditional statements in a HDAE, we associate (informally) a *mode*. For example, if we have ν `if-then-else` statements, we have 2^ν modes, which we can number from 1 to 2^ν.

Currently, we employ the following simple scheme in DAETS. At t_0, fix values for variables and derivatives on which branches in the HDAE depend on. This allows DAETS to select an initial DAE (without branches) to work with. On the first step and after an event is located, we perform the following:

Do at most K iterations of 1–6
1. Perform SA

2. Set values for an initial point
3. Evaluate the mode
4. Project the initial point
5. Evaluate the mode
6. If the mode has not changed, exit from the loop

The DAE is fixed in steps 1 to 4. After the projection in step 4, the projected values may imply a different DAE. If the mode does not change between steps 3 and 5, then we have computed a consistent initial point for a DAE for which the conditions in the branches have not changed. Hence, we accept this DAE and consistent point. Currently, we use $K = 10$, but numerical experiments suggest that generally at most 1 or 2 iterations suffice. If the loop exits and the modes are different, then we cannot find a consistent initial point (see also Sect. 6).

If the mode changes, the new DAE generally has different offsets than the offsets of the previous DAE. Therefore, we may have a different set of variables and derivatives to initialize (see Sect. 5 for an illustration). Since DAETS computes TCs to order much higher than the order of the largest derivative in the DAE, if more values are required to be set in an initial point, we take as initial guesses the corresponding values from the most recent integration step.

4 Model of a Parallel Robot

In the sequel we give the dynamical model of a parallel robot working in closed loop for achieving fast planar, two-degree of freedom (DOF), pick-and-place applications [16]. We also give the equations encoded in DAETS.

4.1 Modeling

Let us consider the parallel robot schematically shown in Fig. 1. It consists of two $\underline{R}RR$ kinematic chains $P_iA_iB_i$ $(i = 1, 2)$ connecting in parallel the traveling plate to the base—R denotes a revolute joint and $\underline{R}$ an actuated revolute joint. The traveling plate is reduced to a point $B = B_1 = B_2$, which is moved in the plane by the two arms of the robot. The rotating arms P_iA_i have length L, whereas l is the length of the forearms A_iB_i. The two actuators are located on the x axis with coordinates $x_p = x_{p_1} = -x_{p_2}$. It is supposed that the task assigned to this 2-DOF robot consists in following prescribed trajectories, such as the one depicted in Fig. 1. We assume that they are given as explicit functions for the position $\mathbf{x}_d(t)$ of point B. The robot's model parameters are given in Table 1.

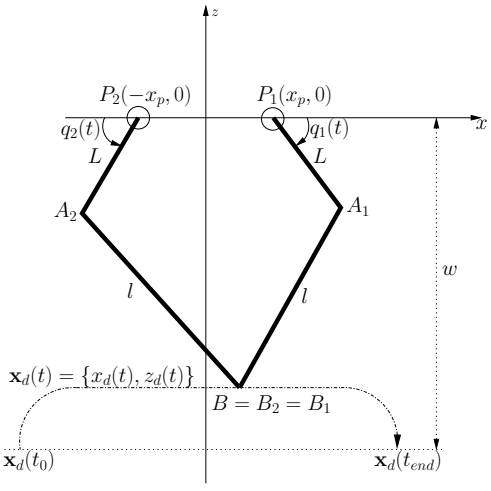

Figure 1: Robot schematics

Table 1: Robot parameters

description		notation	value
Motor (m)	P_1 x-coordinate	x_p	0.1
	P_2 x-coordinate	$-x_p$	0.1
Lengths (m)	P_1A_1, P_2A_2	L	0.3
	A_1B_1, A_2B_2	l	0.7
Masses (kg)	arm	m_1	0.82
	forearm	m_2	0.14
	nacelle	m_3	0.5
	load	m_l	5.0
Inertia (kg.m^2)	motor	J_{mot}	0.37×10^{-4}
	gears	J_{red}	9.09×10^{-4}
	arm	I	0.018895002
friction coefficients		F_s	3.0
		F_v	0.5
maximum delivered torque (N.m)		α	50
gear reduction ratio		η	5
gravity (m/s^2)		g	9.81

4.2 Position equations

Let $\mathbf{x}(t) = \big(x(t), z(t)\big)$ be the coordinates of point B, and let $\mathbf{q}(t) = \big(q_1(t), q_2(t)\big)$ be the vector of P_1 and P_2 articular coordinates, that is of actuated joints coordinates. Position equations are obtained by writing $\|\overrightarrow{A_iB_i}\| = l^2$, where $\overrightarrow{A_iB_i}$ depends on both $\mathbf{x}$ and $\mathbf{q}$. They are

$$0 = \big(x(t) - x_p - L\cos q_1(t)\big)^2 + \big(z(t) + L\sin q_1(t)\big)^2 - l^2 \quad \text{and}$$
$$0 = \big(x(t) + x_p + L\cos q_2(t)\big)^2 + \big(z(t) + L\sin q_2(t)\big)^2 - l^2,$$

which we write as

$$\phi\big(\mathbf{x}(t), \mathbf{q}(t)\big) = 0 \ .$$

4.3 Dynamics

Here we use a simplified modeling for the dynamics of the robot. This is particularly true for the modeling of forearms dynamics, which is done as follows: half the mass of a forearm, $m_2/2$, is added as a point mass at the end of each rotating arm, and a point mass of value $2m_2/2$ is added to the mass of the traveling plate. This assumption may not be verified in general, but it has been validated for the type of pick-and-place tasks considered for this robot [17].

Then, the vector $\mathbf{\Gamma}(t) = \big(\Gamma_1(t), \Gamma_2(t)\big)$, whose components are the torques of the actuators, is given by

$$\mathbf{\Gamma}(t) = \mathbf{\Gamma}_{\text{red}}(t) + \mathbf{\Gamma}_{\text{fric}}(t) + \mathbf{\Gamma}_{\text{arm}}(t) + \mathbf{\Gamma}_{\text{farm}}(t) + \mathbf{\Gamma}_{\text{nac}}(t) \ .$$

where

$$\mathbf{\Gamma}_{\text{red}}(t) = \eta^2 (J_{\text{mot}} + J_{\text{red}})\, \ddot{\mathbf{q}}(t)\,, \tag{4}$$

are the vectors of torques due to the inertia of the motor and gears reflected to the output shaft of the gear box (J_{red} is reflected to the input of the gear box); and

$$\mathbf{\Gamma}_{\text{fric}}(t) = \text{sign}(\dot{\mathbf{q}}(t))\, F_s + F_v\, \dot{\mathbf{q}}(t)\,, \tag{5}$$

are the vectors of equivalent total torques, reflected to the output of the gear box, due to dry and viscous frictions acting between every link. Moreover,

$$\mathbf{\Gamma}_{\rm arm}(t) = I\,\ddot{\mathbf{q}}(t) - m_1 g \cos\mathbf{q}(t)\,, \tag{6}$$

are the vectors of torques due to the inertia and weight of the rotating arm of length L, and

$$\mathbf{\Gamma}_{\rm farm}(t) = 0.5\,m_2\,L\,\left(L\,\ddot{\mathbf{q}}(t) - g\cos\mathbf{q}(t)\right) \tag{7}$$

are the vectors of torques due to the inertia and weight of the forearm. Finally, $\mathbf{\Gamma}_{\rm nac}(t)$, the torques generated by the dynamics of the traveling plate for which $2m_2/2$ has been added, is determined from

$$0 = \left(\mathbf{J}_x^T(t)\mathbf{J}_q^{-1}(t)\right)\mathbf{\Gamma}_{\rm nac}(t) + (m_2 + m_3 + m_l)\left(\ddot{\mathbf{x}}(t) + \mathbf{g}\right)\,, \tag{8}$$

where $\mathbf{J}_q(t) = \partial\phi/\partial\mathbf{q}$ and $\mathbf{J}_x(t) = \partial\phi/\partial\mathbf{x}$. In (5), $\mathrm{sign}(a) = 1$, if $a \geq 0$ and -1 otherwise; in (8), $\mathbf{g} = (0, -g)^T$; and in (4–8), sign, cos, and the differentiation apply componentwise.

4.4 Closed loop control

We are given a prescribed trajectory $\mathbf{x}_d(t) = \left(x_d(t), z_d(t)\right)$. For the robot to track the prescribed trajectory, the joint actuating torques are computed by using a proportional, integral, derivative control in the joint space, with constant gains. Define the tracking error in the joint space by

$$\mathbf{e}(t) = \mathbf{q}_d(t) - \mathbf{q}(t)\,,$$

where $\mathbf{q}_d(t)$ satisfies

$$\phi(\mathbf{x}_d, \mathbf{q}_d) = 0\ .$$

We have

$$\mathbf{\Gamma}_0(t) = K_c\left(\mathbf{e}(t) + \frac{1}{T_i}\int \mathbf{e}(t)dt + T_d\,\dot{\mathbf{e}}(t)\right)\ . \tag{9}$$

Here we use $K_c = 600$, $T_i = 300$ and $T_d = 0.05$. Now, the motor torques $\mathbf{\Gamma}(t) = \left(\Gamma_1(t), \Gamma_2(t)\right)$ are given by, $i = 1, 2$,

$$\Gamma_i(t) = \begin{cases} \alpha & \text{if } \Gamma_{0,i} > \alpha \\ -\alpha & \text{if } \Gamma_{0,i} < -\alpha \\ \Gamma_{0,i}(t) & \text{otherwise}\,, \end{cases}$$

where α is the maximum torque that can be delivered by the actuators.

4.5 DAETS Encoding

Since DAETS does not handle integrals in problem formulation, we introduce two variables $\mathbf{E}(t) = \left(E_1(t), E_2(t)\right)$, write the integral in (9) as

$$0 = \dot{\mathbf{E}}(t) - \mathbf{e}(t),$$

and encode in DAETS

$$0 = \mathbf{\Gamma}_0 - K_c\left(\mathbf{e} + \frac{1}{T_i}\mathbf{E} + T_d\,\dot{\mathbf{e}}\right)\ .$$

(We omit t here and below.) For studying the dynamics of this model, it is convenient to consider Γ_1 and Γ_2 as variables, so we can readily obtain their values. It is also convenient to introduce $\mathbf{y} = (y_1, y_2)^T$ and write $\mathbf{\Gamma}_{\text{nac}} = \mathbf{J}_q\mathbf{y}$. Taking the above considerations into account, we encode in DAETS the second-order system of 12 equations in 12 variables, $\mathbf{x}$, $\mathbf{q}$, $\mathbf{q}_d$, $\mathbf{\Gamma}$, $\mathbf{y}$, and $\mathbf{E}$:

$$0 = \phi(\mathbf{x}, \mathbf{q}) \tag{10}$$

$$0 = \phi(\mathbf{x}_d, \mathbf{q}_d), \quad \mathbf{x}_d \text{ given} \tag{11}$$

$$0 = \mathbf{J}_x^T\mathbf{y} + (m_2 + m_3 + m_l)\,(\ddot{\mathbf{x}} + \mathbf{g}) \tag{12}$$

$$0 = -\mathbf{\Gamma} + \mathbf{\Gamma}_{\text{red}} + \mathbf{\Gamma}_{\text{fric}} + \mathbf{\Gamma}_{\text{arm}} + \mathbf{\Gamma}_{\text{farm}} + \mathbf{J}_q\mathbf{y} \tag{13}$$

$$0 = \begin{cases} \Gamma_i - \alpha & \text{if } \Gamma_{0,i} - \alpha > 0 \\ \Gamma_i + \alpha & \text{if } \Gamma_{0,i} + \alpha < 0, \quad i = 1, 2 \\ \Gamma_i - \Gamma_{0,i} & \text{otherwise} \end{cases} \tag{14}$$

$$0 = \dot{\mathbf{E}} - \mathbf{e} \tag{15}$$

5 Numerical results

We use a prescribed trajectory given by

$$x_d = -0.35\sin(2\pi t) \quad \text{and} \quad z_d = -0.7 + 0.1\cos(2\pi t)$$

and integrate from $t_0 = 0$ to $t_{\text{end}} = 5$.

Initially, we set values for variables on which the branches depend on: $\mathbf{q} = (0, 0)$, $\mathbf{q}' = (0, 0)$, $\mathbf{q}_d = (0, 0)$, and $\mathbf{q}'_d = (0, 0)$. These values imply $\mathbf{\Gamma}_{\text{fric}} = F_s + F_v\dot{\mathbf{q}}$ in (5), and correspondingly in (13), and $0 = \mathbf{\Gamma} - \mathbf{\Gamma}_0$ in (14). The SA determines that the resulting DAE is quasilinear, of index 3, and initial values must be given for $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{q}$, $\mathbf{q}'$, $\mathbf{q}_d$, and $\mathbf{E}$. We set, in addition to the above values, $\mathbf{x} = (0, -0.6)$, $\mathbf{x}' = (0, 0)$, and $\mathbf{E} = (0, 0)$.

DAETS applies the method from §3.2 and finds, and t_0, a not quasilinear system of index 3 that requires initial values for $\mathbf{x}$, $\dot{\mathbf{x}}$, $\ddot{\mathbf{x}}$, $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, $\mathbf{q}_d$, $\mathbf{y}$, $\mathbf{\Gamma}$, $\mathbf{E}$, and $\dot{\mathbf{E}}$. The solver determines initial values for these variables and finds a consistent initial point for the resulting DAE.

In Fig. 2, we display the signature matrices of the DAEs at t_0. From the rows corresponding to equation (14), the new DAE takes constant values for Γ_1, and Γ_2. During the integration from t_0 to t_{end}, (10–15) switches between 18 different modes, out of 2^6 possible modes. The signature matrix changes only in rows 9 and 10, corresponding to (14).

In Fig. 3, we show the prescribed and computed trajectories in x-z space, and also Γ_1 and Γ_2 versus t. In Fig. 4, we show the prescribed and computed by DAETS trajectories versus t. Their derivatives are shown in Fig. 5, where the located events in $\mathbf{q}'$ are also displayed. Finally, plots of $\Gamma_{0,1} \pm \alpha$, $i = 1, 2$, versus t, along with their located zeros, are given in Fig. 6.

DAETS determined successfully all the events in this integration interval, and found the correct DAEs and consistent initial points. It also kept $\Gamma_i \in [-\alpha, \alpha] = [-50, 50]$, $i = 1, 2$. The integration time is ≈ 0.65 seconds on Mac OS X, 2.8GHz Intel dual-core processor with 2GB RAM, and the GCC compiler.

	x	z	q_1	q_2	q_{d_1}	q_{d_2}	y_1	y_2	Γ_1	Γ_2	E_1	E_2	c_i
(10)	0∗	0	0	–	–	–	–	–	–	–	–	–	2
	0	0∗	–	0	–	–	–	–	–	–	–	–	2
(11)	–	–	–	–	0∗	–	–	–	–	–	–	–	1
	–	–	–	–	–	0∗	–	–	–	–	–	–	1
(12)	2	–	0	0	–	–	0	0∗	–	–	–	–	0
	–	2	0	0	–	–	0∗	0	–	–	–	–	0
(13)	0	0	2∗	–	–	–	0	–	0	–	–	–	0
	0	0	–	2∗	–	–	–	0	–	0	–	–	0
(14)	–	–	1	–	1	–	–	–	0∗	–	0	–	0
	–	–	–	1	–	1	–	–	–	0∗	–	0	0
(15)	–	–	0	–	0	–	–	–	–	–	1∗	–	0
	–	–	–	0	–	0	–	–	–	–	–	1∗	0
d_j	2	2	2	2	1	1	0	0	0	0	1	1	

	x	z	q_1	q_2	q_{d_1}	q_{d_2}	y_1	y_2	Γ_1	Γ_2	E_1	E_2	c_i
	0∗	0	0	–	–	–	–	–	–	–	–	–	2
	0	0∗	–	0	–	–	–	–	–	–	–	–	2
	–	–	–	–	0∗	–	–	–	–	–	–	–	0
	–	–	–	–	–	0∗	–	–	–	–	–	–	0
	2	–	0	0	–	–	0	0∗	–	–	–	–	0
	–	2	0	0	–	–	0∗	0	–	–	–	–	0
	0	0	2∗	–	–	–	0	–	0	–	–	–	0
	0	0	–	2∗	–	–	–	0	–	0	–	–	0
	–	–	–	–	–	–	–	–	0∗	–	–	–	0
	–	–	–	–	–	–	–	–	–	0∗	–	–	0
	–	–	0	–	0	–	–	–	–	–	1∗	–	0
	–	–	–	0	–	0	–	–	–	–	–	1∗	0
d_j	2	2	2	2	0	0	0	0	0	0	1	1	

Figure 2: Signature matrices of the initial DAE at t_0 (top) and the DAE found by DAETS at t_0 (bottom). The numbers in brackets correspond to the equations in (10–15).

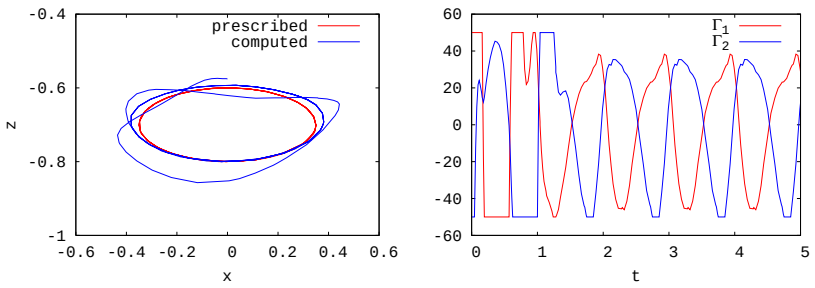

Figure 3: Plots of prescribed and computed trajectories (left) and computed Γ_1 and Γ_2 (right).

6 Conclusion

With DAETS, we can integrate systems containing DAEs in a compact form: high-index, any order, and fully nonlinear in highest order derivatives. Thus, when a problem is modeled by a DAE, or HDAE, we can write equations in their "natural" form, without further transformations.

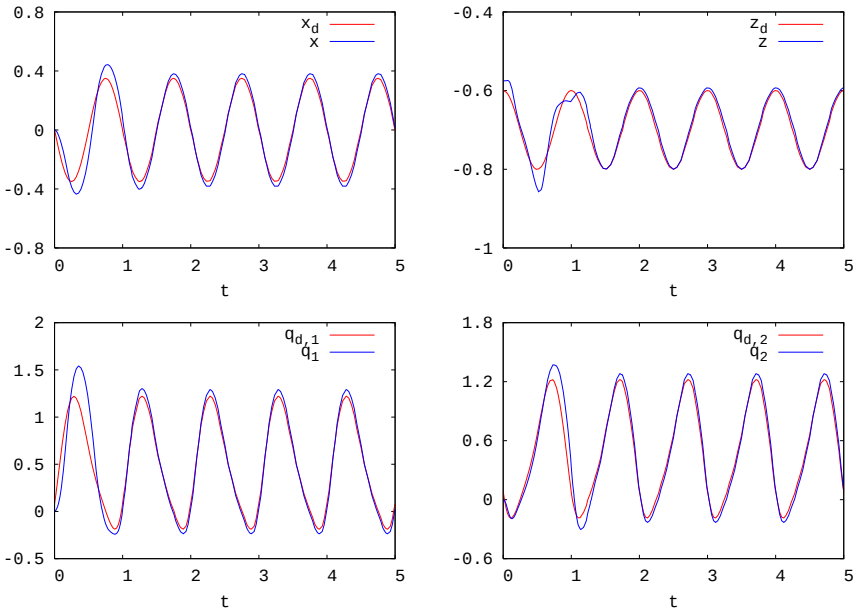

Figure 4: Plots of prescribed $\mathbf{x}_d$ and computed $\mathbf{x}$, $\mathbf{q}_d$, and $\mathbf{q}$.

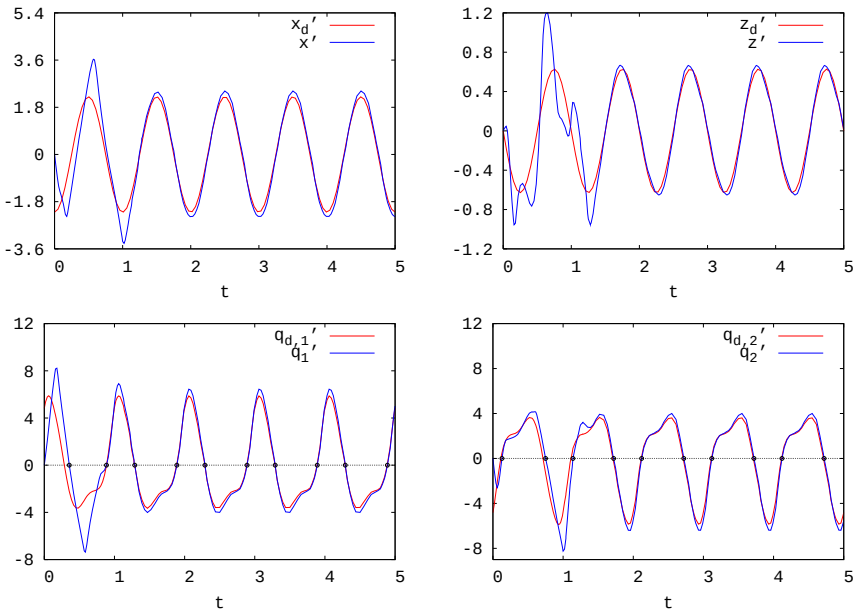

Figure 5: Plots of prescribed $\mathbf{x}'_d$ and computed $\mathbf{x}'$, $\mathbf{q}'_d$, and $\mathbf{q}'$. The located events in $\mathbf{q}'$ are marked with points.

DAETS has proved very effective on DAEs with sufficiently differentiable f_i, and we have shown promising results towards integrating HDAEs. We do not have to generate DAEs in advance: they are discovered at runtime when events

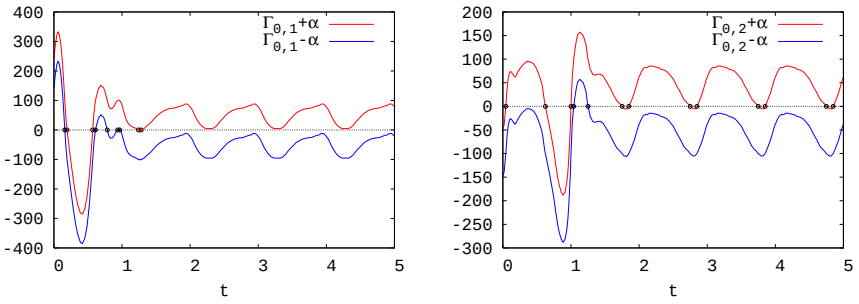

Figure 6: Plots of $\Gamma_{0,i} \pm \alpha$ with located events.

are detected. This entails little overhead, as when a DAE occurs more than once, the SA analysis is redone for the same structure of a DAE. This leads to mild inefficiency, since the work for SA is small compared to the work for overall integration. One possibility for avoiding repetition of the SA, for the same DAE, is to keep a database of sigma matrices and offsets for each DAE discovered during integration.

Our simple method for consistent initialization does not always lead to discovering the correct DAE and finding a consistent point (cf. [9]). Future work will involve adding a robust method for consistent initialization. Namely, when the present method does not succeed, we plan to invoke a more robust algorithm, likely a continuation-based method, similar to the one presented in [9]. Here, we should point out that DAETS has proved accurate and robust when solving pure algebraic systems through continuation [3].

Finally, future work will include detailed numerical analysis of event location and consistent initialization. We aim to incorporate these features in the next release of DAETS.

References

[1] Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (I): Computing Taylor coefficients. BIT **45**(3) (2005) 561–591

[2] Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (II): Computing the System Jacobian. BIT **47**(1) (2007) 121–135

[3] Nedialkov, N.S., Pryce, J.D.: Solving differential-algebraic equations by Taylor series (III): The DAETS code. J. Numerical Analysis, Industrial and Applied Mathematics **3**(1–2) (2008) 61–80

[4] Nedialkov, N., Pryce, J.D.: DAETS user guide. Technical Report CAS-08-08-NN, Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada, L8S 4K1 (2008)

[5] Brenan, K., Campbell, S., Petzold, L.: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. Second edn. SIAM, Philadelphia (1996)

[6] Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer Verlag, Berlin (1991)

[7] Pryce, J.D.: A simple structural analysis method for DAEs. BIT **41**(2) (2001) 364–394

[8] Hamann, P., Mehrmann, V.: Numerical solution of hybrid systems of differential-algebraic equations. Comput. Methods Appl. Mech. Eng. **197**(6–8) (2008) 693–705

[9] Najafi, M., Nikoukhah, R., Campbell, S.: Computation of consistent initial conditions for multi-mode DAEs: application to Scicos. In: IEEE International Symposium on computer aided control systems. Volume 4., IEEE, Piscataway, NJ (2004) 131–136

[10] Pryce, J.D.: Solving high-index DAEs by Taylor Series. Numerical Algorithms **19** (1998) 195–211

[11] Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. Frontiers in applied mathematics. SIAM, Philadelphia, PA (2000)

[12] Stauning, O., Bendtsen, C.: FADBAD++ web page

[13] Wächter, A.: An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering. PhD thesis, Carnegie Mellon University, Pittsburgh, PA (2002)

[14] Park, T., Barton, P.I.: State event location in differential-algebraic models. ACM Trans. Model. Comput. Simul. **6**(2) (1996) 137–165

[15] Bini, D.A.: Numerical computation of polynomial zeros by means of Aberth's method. Numerical Algorithms **13**(2)

[16] Ramdani, N., Gouttefarde, M., Pierrot, F., Merlet, J.: First results on the design of high speed parallel robots in presence of uncertainty. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2008) 2410–2415

[17] Nabat, V., Rodriguez, M., Company, O., Pierrot, F., Dauchez, P.: Very fast schoenflies motion generator. In: IEEE Int. Conf. on Industrial Technology (ICIT). (2005) 365–370

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399